A Brief on Constraint Solving

Christoph M. Hoffmann[1] and Robert Joan-Arinyo[2]

[1]Purdue University, cmh@cs.purdue.edu

[2]Universitat Politècnica de Catalunya, robert@lsi.upc.edu

ABSTRACT

We survey the current state of the art in geometric constraint solving. 2D constraint solving is considered, and different approaches are characterized.

Keywords: Geometric constraints, constraint solving, parametric design.

1. INTRODUCTION AND SCOPE

2D geometric constraint solving is arguably a core technology of computer-aided design (CAD) and, by extension, of managing product design data. Since the introduction of parametric design by Pro/Engineer in the 1980s, every major CAD system has adopted geometric constraint solving into its design interface. Most prominently, 2D constraint solving has become an integral component of sketchers on which most systems base feature design.

Beyond applications in CAD and, by extension, in manufacturing, geometric constraint solving is also applicable in virtual reality and is closely related in a technical sense to geometric theorem proving. For solution techniques, geometric constraint solving also borrows heavily from symbolic algebraic computation and matroid theory.

In this paper, we review many of the basic techniques that are widely available for solving 2D and 3D geometric constraint problems. We focus primarily on the basics of 2D solving and touch lightly on spatial constraint solving and the various ways in which geometric constraint solvers can be extended with relations, external variables, and parameter value enclosures. These and other extensions and problem variants have been published in the literature. They are recommended to the interested reader as follow-on material for study.

This review has been abbreviated to fit required page limitations. An expanded version of this survey can be downloaded at http://www.cs.purdue.edu/homes/cmh/distribution/papers/Constraints/ThailandFull.pdf or at http://www.lsi.upc.es/~robert/ThailandFull.pdf.

2. PRELIMINARIES

A geometric constraint problem can be characterized by means of a tuple (E, O, X, C) where

- E is the geometric space constituting a reference framework into which the problem is embedded. E is usually Euclidean.
- O is the set of specific geometric objects which define the problem. They are chosen from a fixed repertoire including points, lines, circles and the like.
- X is a, possibly empty, set of variables whose values must be determined. In general, variables represent quantities with geometric meaning: distances, angles and so on. When the quantities are without a geometric meaning, for example, when they quantify technological aspects and functional capabilities, those variables are called *external.*
- C is the set of constraints. Constraints can be geometric or equational. Geometric constraints are relationships between geometric elements chosen from a predefined set, e.g., distance, angle, tangency, etc.

The relationship (the distance, the angle ...) is represented by a tag. If the tag represents a fixed value, known in advance, then the constraint is called *valuated*. If the tag represents a value to be computed as part of solving the constraint problem, then the constraint is called symbolic. Equational constraints are equations some of whose variables are tags of symbolic constraints. The set of equational constraints can be empty.

The geometric constraint solving problem can now be stated as follows:

Given a geometric constraint problem (E, O, X, C),

1. Are the geometric elements in O placed with respect to each other in such a way that the constraints in C are satisfied? If the answer is positive, then
2. given an assignment of values to the valuated constraints and external variables, is there an actual construction that satisfies the constraints and equations?

When dealing with geometric constraint solving, the first issue that needs to be settled is the dimension of the embedding space E. In 2D Euclidean space, $E = \mathbf{R}^2$, a number of techniques have been developed that successfully solve the geometric constraint solving problem. For an in-depth review see Jermann, [19]. However, there remain open questions such as characterizing the competence (also called domain) of the known techniques.

Spatial constraint solving, where $E = \mathbf{R}^3$, include problems in fields like molecular modeling, robotics, and terrain modeling. Here, both a good conceptualization and an effective solving methodology for the geometric constraint problem has proved to be difficult. Pioneering work has been reported by Hoffmann and Vermeer, [12, 13] and by Durand, [6].

2.1 The General Problem

The figure depicts a general geometric constraint solving problem in $\mathbf{R}^2$. Problem components are

- The set of geometric elements, $O = \{A, B, C, D, L_{AB}, L_{AC}, L_{BC}\}$.
- The set of tags in the constraints, $P = \{d, h, \alpha\}$.
- The set of geometric variables, $V_1 = \{x, y\}$.
- The set of external variables, $V_2 = \{v\}$.
- The set of geometric constraints with fixed tags, $C_1 = \{dpp(A,B) = d,\ dpl(C, L_{AB}) = h,\ ang(L_{AB}, L_{BC}) = \alpha\}$.
- The set of geometric constraints with variable tags, $C_2 = \{dpp(A,C) = x,\ dpp(C,D) = y\}$.
- The set of equational constraints, $C_3 = \{y = x \cdot v,\ v = 0.5 \cos(\alpha)\}$.

with $X = V_1 \cup V_2$ and $C = C_1 \cup C_2 \cup C_3$.

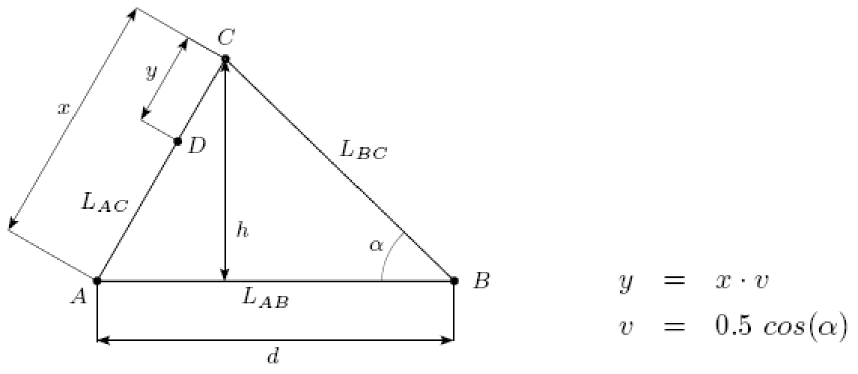

Fig. 1: The general geometric constraint solving problem; example in $\mathbf{R}^2$

Presented in this way, the geometric constraint solving problem includes in general issues concerning how to deal with external variables. Here we refer the interested reader to the work by Hoffmann and Joan-Arinyo, [11], and Joan-Arinyo and Soto, [20].

2.2 The Basic Problem

The basic constraint problem only considers geometric elements and constraints whose tags are assigned a value. It excludes external variables, constraints whose tags must be computed, and equational constraints. So the basic problem is stated in the following way.

Given a set O with n geometric elements and a set C with m geometric constraints defined on them

- Is there a placement of the n geometric elements such that the m constraints are fulfilled? If the answer is positive,
- given an assignment of values to the m constraints tags, is there an actual construction of the n geometric elements satisfying the constraints?

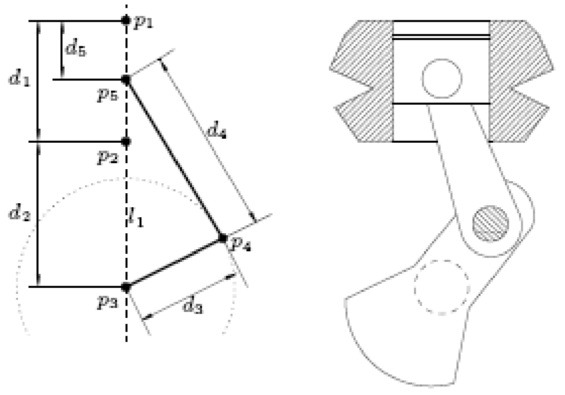

Fig. 2. Piston, crankshaft and connecting rod mechanism

Figure 2 depicts a piston-crankshaft mechanism, a basic geometric constraint solving problem. The left side shows the geometric problem, the right side shows the actual mechanism so abstracted. The mechanism transforms the translational motion of point p_5 along the straight line l_1 into a rotational motion of point p_4, on a circular path with center p_3 and radius d_3. The piston-crankshaft mechanism can be abstracted as a geometric constraint solving problem comprising five points p_i, $1 \leq i \leq 5$, and a straight line l_1. The set of constraints is as shown in the drawing.

2.3 Problem Categorization

The CAD/CAM community focuses on the design and manufacture of rigid objects, that is, objects that are fully determined up to a global coordinate system. Similarly, we seek solutions to a constraint problem that are determined up to a global coordinate system, that is, where solutions are congruent under the rigid-body transformations of translation and rotation. We call a configuration of geometric objects in Euclidean space *rigid* when all objects are fixed with respect to each other up to translation and rotation.

An intuitive way to introduce rigidity comes from considering the number of solutions that a geometric constraint problem has. There are three categories:

A problem is *structurally under constrained* if there are infinitely many solutions that are not congruent under rigid transformation, *structurally well-constrained*, if there are finitely many solutions modulo rigid transformation, and *structurally over constrained* if the deletion of one or more constraints results in a well-constrained problem. A constraint problem naturally corresponds to a set of (usually nonlinear) algebraic equations.

Defined in this way, the concept of rigidity appears to be simple but it is not quite in accord with the intuition about rigidity. The categories so defined only refer to the problem's structure and do not account for other issues such as inconsistencies that could originate from specific values assigned to the constraints. Clearly a problem that is structurally well-constrained could actually be underconstrained for specific values of the constraints.

For example, consider the *structurally* well constrained problem given in Figure 3, see Fudos and Hoffmann, [10]. Point P is properly placed whenever $\alpha+\beta\neq 90^\circ$ and the problem is well-constrained. But if $\alpha + \beta = 90^\circ$, then the placement for point P is undetermined and, therefore, the problem is no longer well constrained.

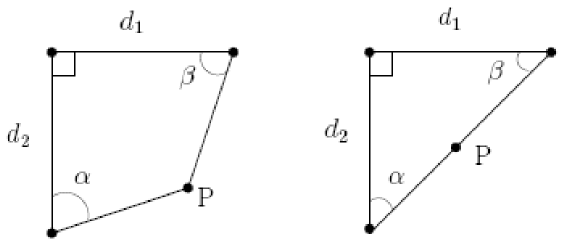

Fig. 3. Left, general configuration; right: degenerate configuration for $\alpha+\beta=90°$

3. GENERAL PROPERTIES OF SOLVING TECHNIQUES

The general properties a constraint solver should have include the following:

- **Soundness**: The solver always halts and, if a solution is found, it satisfies all stipulated constraints.
- **Completeness**: The solver solves all solvable problems and announces the unsolvability of all unsolvable problems.
- **Competence**: Since the known, efficient methods are not complete, property that is more attractive in practice is that the solver solves all solvable problems in a subdomain of interest and, for unsolvable problems in the subdomain, announces unsolvability. Note that a characterization of the subdomain may be difficult.
- **Persistence**: With the same set of (valuated) constraints the solver always finds the same solution.
- **Stability**: Under small changes of assignments to the geometric constraint tags, the solution found is nearby.
- **Efficiency**: Efficiency measures the computational cost, that is, whether solutions are found rapidly.
- **Robustness**: A solver is robust if the solutions are not adversely affected by the finite precision of floating-point computations.
- **Intensionality**: The solver finds solutions that the user is interested in.
- **Geometric sense**: A solver has geometric sense whenever the solutions it finds can be expressed as a sequence of geometric construction steps.
- **Dimension independence**: The solver can be used for problems embedded in the space $E = \mathbf{R}^n$ independently of n.
- **Generality**: The solver can deal with the general geometric constraint solving problem including symbolic constraints and external variables.

For 2D solvers, there are good compromises that achieve these solver characteristics to a reasonable degree. However, it is difficult to obtain solvers that exhibit persistence and stability fully, [23].

In general, solver competence is the antithesis of efficiency since constraint solving is of doubly exponential complexity. Intensional problems include root selection, orientation, and topological degeneracy for specific dimensional values. They may look simple but generally are associated with complex mathematical problems.

4. MAJOR APPROACHES

Geometric constraint solving methods can be roughly classified as graph-based, logic-based, or algebraic. For 2D solvers, the graph-based approach has become dominant in CAD. A problem closely related to geometric constraint solving is Automated Theorem Proving.

4.1 Graph-Based Approaches

In the graph-based approach, the constraint problem is translated into a graph (or hyper graph) whose vertices represent the geometric elements and whose edges the constraints upon them. The solver analyzes the graph and formulates a solution strategy whereby subproblems are isolated and their solutions suitably combined. A subsequent phase then solves all subproblems and combines them. The advantage of this type of solver is that the subproblems often are very small and fall into a few simple categories. The disadvantage is that the graph analysis of a fully competent solver is rather complicated.

The graph-based approach can be further subdivided into constructive, degree of freedom analysis, and propagation.

4.1.1 Constructive Approach

The constructive approach generates the solution to a geometric constraint problem as a symbolic sequence of basic construction steps. Each step is a rule taken form a predefined set of operations that position a subset of the geometric elements. For example, the operations may restrict to ruler-and-compass constructions. Clearly, this approach preserves the geometric sense of each operation involved in the solution. Note that the sequence of construction steps allows to compactly represent a possibly exponential number of solution instances. However, the constructive approach cannot solve problems with symbolic constraints or external variables.

Depending on the technique used to analyze the problem, two different categories of constructive approaches can be distinguished: *top-down* and *bottom-up*.

The top-down technique recursively splits the problem until it has isolated simpler, basic problems whose solutions are known. In this category, Todd, [33], defines the *r-tree* concept and derives a geometric constraint solving algorithm. Owen, [31], describes a more general method based on the recursive decomposition of the constraints graph into triconnected components. Inspired by Owen's work, Fudos reported a new decomposition method in [8]. Efficient algorithms with a running time $O(n^2)$ where n is the number of geometric elements, are known for the methods of Owen and Fudos.

In the bottom-up approach, the solution is built by suitably combining recursively solutions to subproblems already computed, starting from the constraints in the given set, considering each constraint as a single element.

Constraints may be represented implicitly as a collection of sets of geometric elements where the elements of each set are placed with respect to a local framework. Sets are merged; e.g., by application of rewriting rules until all the geometric elements are included in just one set. The advantage of this representation is that the sets of constraints capture the relationships between geometric elements compactly. Fudos *et al.*, in the method described in [4, 9, 10], use one type of sets of constraints, called *cluster*, and one generic rule that merges three clusters which pairwise share two elements.

Lee *et al.*, [29], describe a constructive method that associates with each vertex in the graph a status which can be defined, half defined or not defined. Inference rules are used to modify the status of the vertices.

Efficient algorithms with a running time $O(n^2)$, where n is the number of geometric elements, are known for the methods listed above. However, the constructive approach is not complete, therefore assessing the competence of solvers in this category is an important issue. Verroust, [35, 36], partially characterizes the set of relevant problems solved by the solver described. Joan-Arinyo *et al.*, [21, 22], describe a formalization that unifies the methods reported by Fudos, [9, 10], and Owen, [31]. In [21], Joan-Arinyo *et al.* show that the sets of problems solved by Fudos' and Owen's approaches are the same.

In [11] Hoffmann and Joan-Arinyo describe a technique that extends constructive methods with the capability of managing functional relationships between geometric and externals variables. Essentially, the technique combines two methods: one is a constructive constraint solving method, and the other is a systems of equations analysis method. Joan-Arinyo and Soto-Riera, [20], further improved the technique and formalized it as a rewriting system.

Constructive methods work well in 2-space. Several attempts to extend them to 3-space have been reported. See, for example, Brüderlin [5], Verroust [35], and Hoffmann and Vermeer [12, 13].

4.1.2 Degrees of Freedom Analysis

Degrees of Freedom Analysis assigns degrees of freedom to the geometric elements by labeling the vertices of the graph of the problem. Each edge of the graph is labeled with the number of degrees of freedom canceled by the associated constraint. Then the method solves the problem by analyzing the resulting labeled graph.

Kramer, [24, 25, 26], developed a method to solve specific problems from the kinematics of mechanisms.

The method applies techniques borrowed from the process planning field to yield a symbolic solution. Since the set of rules used to generate the plan preserves geometric sense, the entire method also preserves it. Kramer, proves that his method is correct by showing that the set of rules together with the labeled graph is a canonical rewriting system. The method runs in time O(n m), where n is the number of geometric elements and m the number of constraints in the problem. Since m is typically O(n), the method has the same complexity as the constructive approach. Bhansali *et al.*, [1], describe a method that generates automatically segments of the symbolic solution in Kramer's approach.

In [17, 18], Hsu reports a method with two phases. First, a symbolic solution is generated. Then, the actual construction is carried out. The method applies geometric reasoning and, if this fails, numerical computation.

Latham *et al.*, [28], decompose the labeled graph in minimal connected components called *balanced sets*. If a balanced set corresponds to one of the predefined specific geometric constructions, then it can be solved. Otherwise the underlying equations are solved numerically. The method also deals with symbolic constraints and identifies over- and under constrained problems. Assigning priorities to the constraints allows them to solve over constrained problems. A proof of correctness is also given.

Hoffmann *et al.*, [14, 15, 16], have developed a flow-based method for decomposing graphs of geometric constraint problems. The method generically iterates to obtain a decomposition of the underlying algebraic system into small subsystems called *minimal dense* subgraphs. The method fully generalizes degree-of-freedom calculations, the approaches based on matching specific subgraphs patterns, as well as the prior flow-based approaches. However, the decomposition rendered does not necessarily have geometric sense since minimal dense subgraphs can be of arbitrary complexity far exceeding problems that yield to classical geometric construction.

4.1.3 Propagation Approach

Propagation methods represent the set of algebraic equations with a symmetric graph whose vertices are variables and equations and whose edges are labeled with the occurrences of the variables in the equations.

Propagation methods try to orient the edges in the graph in such a way that each equation vertex is a sink for all edges incident on it except one. If the process succeeds, then there is a general incremental solution. That is, the system of equation can be transformed into a triangular system and solved using back substitution.

Among the techniques to orient a graph we find in the literature degrees of freedom propagation and propagation of known values, [7, 32, 34].

Propagation methods do not guarantee finding a solution whenever one exists. They fail when the orientation algorithm finds a loop. Propagation methods can combined with numerical methods for equation solving to ameliorate circularity, [2, 27]. Veltkamp and Arbab, [34], apply other techniques to break loops created while orienting the graph.

Leler, [30], describes propagation methods in depth and proposes *augmented rewriting terms*, a tool which consists of a classical rewriting system along with an association of atomic-value and object-type. This tool has had success in solving certain systems of nonlinear equations.

In [3], Borning *et al.*, describe an local propagation algorithm that can deal with inequalities.

4.2 Logic-Based Approach

We review this approach without citations, referring the interested reader to the expanded version of this paper available on the web as pointed out in the introduction.

In the logic-based approach, the problem is translated into a set of assertions and axioms characterizing the constraints and the geometric objects. By employing reasoning steps, the assertions are transformed in ways that expose solution steps in a stereotypical way and special solvers then compute coordinate assignments.

Aldefeld, Brüderlin, Sohrt, and Yamaguchi *et al.*, use first order logic to derive geometric information applying a set of axioms from Hilbert's geometry. Essentially these methods yield geometric loci at which the elements must be.

Sunde, and Verroust, consider two different types of sets of constraints: sets of points placed with respect to a local framework, and sets of straight line segments whose directions are fixed with respect to a local framework. The reasoning is basically performed by means of a rewriting system on the sets of constraints. The problem is solved when all the geometric elements belong to a unique set.

Joan-Arinyo and Soto-Riera extended these sets of constraints with a third type consisting of sets containing one point and one straight line such that the perpendicular point-line distance is fixed.

4.3 Algebraic Methods

We review this approach without citations, referring the interested reader to the expanded version of this paper available on the web as pointed out in the introduction.

In the algebraic approach, the constraint problem is translated directly into a set of nonlinear equations and is solved using any of the available methods for solving nonlinear equations. The main advantages of algebraic solvers are their generality, dimension independence and the ability to deal with symbolic constraints naturally.

In principle, an algebraic solver can be fully competent. However, algebraic solvers may have low efficiency or may have difficulty constructing solutions reliably. When used to pre-process and study specific constraint systems, however, algebraic techniques can be extremely useful and very practical.

As a result of mapping the geometric domain problem into an equational one, the geometric sense of the solutions rendered is lost. Moreover, well constrained problems are mapped to under constrained systems of equations because constraints fix the placement for each geometric element with respect each other only modulo translation and rotation. Therefore a set of additional equations must be joined to cancel these remaining degrees of freedom.

Algebraic methods can be further classified according to the specific technique used to solve the system of equations, namely into numerical, symbolic, and analysis of systems of equations.

4.3.1 Numerical Methods

Numerical methods provide powerful tools to solve iteratively large systems of equations. In general, a good approximation of the intended solution should be supplied to guarantee convergence. This means that if, as it is customary, the starting point is taken from the sketch defined by the user, then the sketch should be close to the intended solution. The numerical methods may offer little control over the solution in which the user is interested. To achieve robustness, numerical iterative methods must be carefully designed and implemented.

Borning, Hillary and Braid, and Sutherland, use a *relaxation* method. This method is an alternative to the propagation method. Basically, the method perturbs the values assigned to the variables and minimizes some measure of the global error. In general, convergence to a solution is slow.

The method most widely used is the well-known *Newton-Raphson iteration*. It is used in many solvers. Newton-Raphson is a local method and converges much faster than relaxation. The method does not apply to consistently over constrained systems of equations unless special provisions are made such as combining it with least-squares techniques.

Homotopy or *continuation* is a family of methods with a growing popularity. These methods are global and guarantee convergence. Moreover, they are exhaustive and allow to determine all solutions of a constraint problem. However, their efficiency is worse than that of Newton-Raphson. Lamure and Michelucci, and Durand, apply this method to geometric constraint solving.

Other, less conventional methods have also been proposed. For example, Hel-Or *et al.* introduced the *relaxed parametric design* method where the constraints are soft, that is, they do not have to be met exactly, the problem is modeled as a static stochastic process, and the resulting system of probabilistic equations is solved using the Kalman filter familiar from control theory. The Kalman filter was developed to efficiently compute linear estimators and when applied to nonlinear systems, it does not necessarily finds a solution even if one exists.

Kin *et al.*, reported on a numerical method based on extended Boltzmann machines which are a sort of neural network whose goal is to minimize a given polynomial that measures the energy of the system.

4.3.2 Symbolic Methods

Symbolic algebraic methods compute a Gröbner basis for the given system of equations. Algorithms to compute these bases include those by Buchberger, and by Wu-Ritt. These methods, essentially, transform the system of polynomial equations into a triangular system whose solutions are those of the given system. In effect, triangularization reduces solving a simultaneous, nonlinear system to univariate root finding. Forward or a backward substitution must be used.

Buchanan *et al.*, describe a solver built on top of the Buchberger's algorithm. Kondo reports a symbolic algebraic method and later improves that work by generating a polynomial that summarizes the changes undergone by the system of equations.

4.3.3 Analysis of Systems of Equations

Methods based on the analysis of systems of equations determine whether a system is under-, well- or over-constrained from the system structure. These methods can be extended to decompose systems of equations into a set of minimal graphs which can be solved independently. They can be used as a pre-processing phase for any other method, reducing the number of variables and equations that must be solved simultaneously.

Serrano applies analysis of systems of equations to select from a set of candidate constraints a well constrained, solvable subsets of equations.

4.4 Theorem Proving

Solving a geometric constraint problem can be seen as automatically proving a geometric theorem. However, automatic geometric theorem proving requires more general techniques and, therefore, methods which are much more complex than those required by geometric constraint solving.

Wu Wen Tsün pioneered the Wu-Ritt method, an algebraic-based geometric constraint solving method. In [37, 38], he uses it to prove geometric theorems. The method automatically finds necessary conditions to obtain non-degenerated solutions. Chou applies Wu's method to prove novel geometric theorems. He reports on a work in automatic geometric theorem proving which allows to interpret, from a geometric point of view, the proof generated by computation.

ACKNOWLEDGEMENTS

We wish to acknowledge our collaborators over the years, especially William Bouma, Jiazen Cai, Xiangping Chen, Ching-Shoei Chiang, Cassiano Durand, Ioannis Fudos, Xiaoshan Gao, Ku-Jin Kim, Ramanathan Kavasseri, Andrew Lomosov, Robert Paige, Meera Sitharam, Antoni Soto, Pamela Vermeer, Sebastià Vila, Weiqiang Yang, and Bo Yuan. Much of the material reviewed has been to their credit.

R. Joan-Arinyo has been supported by The Ministerio de Educaciòn y Ciencia and by FEDER under grant TIN2004-06326-C03-01. Hoffmann's work has been supported in part by NSF grants DMS-013098, DCNS-0216131, DHER-0227828, DSC-0325227, DCMS-0443148, and by an IBM faculty scholar award.

REFERENCES

[1] S. Bhansali, G. Kramer, and T.J. Hoar. A principled approach towards symbolic geometric constraint satisfaction. *Journal of Artificial Intelligence Research*, 4:419-443, 1996.

[2] A. H. Borning. The programming language aspects of ThingLab, a constrained oriented simulation laboratory. *ACM Trans. on Prog. Lang. and Systems*, 3(4):353-387, 1981.

[3] A. Borning, R. Anderson, and B. Freeman-Benson. Indigo: A local propagation algorithm for inequality constraints. In UIST'96, pages 129--136, Seattle, Washington, USA, November 6-8 1996. ACM.

[4] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. A geometric constraint solver. *Computer Aided Design*, 27(6):487-501, 1995.

[5] B. D. Brüderlin. Constructing three-dimensional geometric objects defined by constraints. *Workshop on Interactive 3D Graphics*, pages 111-129. ACM, October 1986.

[6] C. Durand. *Symbolic and numerical techniques for constraint solving*. PhD thesis, Comp. Sci., Purdue Univ., 1998.

[7] B. Freeman-Benson, J. Maloney, and A. Borning. An incremental constraint solver. *Comm. ACM*, 33(1):54-63, 1990.

[8] I. Fudos. *Constraint Solving for Computer Aided Design*. PhD thesis, Purdue University, Department of Computer Sciences, 1995.

[9] I. Fudos and C. M. Hoffmann. Correctness proof of a geometric constraint solver. *Intl J Computational Geometry and Applications*, 6(4):405-420, 1996.

[10] I. Fudos and C. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Trans. On Graphics 16,* p.179-216, 1997.

[11] C. M. Hoffmann and R. Joan-Arinyo. Symbolic constraints in constructive geometric constraint solving. *J. Symbolic Comp. 23,* p.287-300, 1997.

[12] C. M. Hoffmann and P. J. Vermeer. Geometric constraint solving in $\mathbf{R}^2$ and $\mathbf{R}^3$. In D.-Z. Du and F. Hwang, editors, *Computing in Euclidean Geometry,* p.266-298. World Scientific Publishing, 1995.

[13] C. M. Hoffmann and P. J. Vermeer. A spatial constraint problem. In J. P. Merlet and B. Ravani, editors, *Computaional Kinematics '95,* p./83-92. Kluwer Academic Publishers, 1995.

[14] C. M. Hoffmann, A. Lomonosov, and M. Sitharam. Finding solvable subsets of constraint graphs. In Principles and Practice of Constraint Programming, 463-477, Schloss Hagenberg, Austria, 1977.

[15] C. M. Hoffmann, A. Lomonosov, and M. Sitharam. Decompostion Plans for Geometric Constraint Problems, Part I: Performance Measurements for CAD. *Journal of Symbolic Computation*, 31:367--408, 2001.

[16] C. M. Hoffmann, A. Lomonosov, and M. Sitharam. Decompostion Plans for Geometric Constraint Problems, Part II: New Algorithms. *Journal of Symbolic Computation*, 31:409-427, 2001.

[17] C.-Y. Hsu. *Graph-Based Approach for Solving Geometric Constraint Problems.* PhD thesis, Department of Computer Science. The University of Utah, June 1996.

[18] C.-Y. Hsu and B. D. Brüderlin. A hybrid constraint solver using exact and iterative geometric constructions. In D. Roller and P. Brunet, eds, *CAD Systems Development: Tools and Methods*, 265-279, 1997. Springer-Verlag.

[19] C. Jermann. Résolution de constraints géométriques par rigidification récursive et propagation d'intervalles. PhD thesis, Université de Nice-Sophia Antipolis, 2002.

[20] R. Joan-Arinyo and A. Soto-Riera. Combining constructive and geometric constraint solving techniques. *ACM Trans. On Graphics 18,* p.35-55, 1999.

[21] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana. On the domain of constructive geometric constraint solving techniques. In R. Duricovic and S. Czanner, editors, *Spring Conference on Computer Graphics*, pages 49--54, Budmerice, Slovakia, April 25-28 2001. IEEE Computer Society, Los Alamitos, CA.

[22] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, and J. Vilaplana. Declarative characterization of a general architecture for constructive geometric constraint solvers. In D.~Plemenos, editor, *Fifth Intl Conf on Computer Graphics and Artificial Intelligence*, pages 63-76, Limoges, France, May 2002. Universitè de Limoges.

[23] U. Kortenkamp. *Foundations of Dynamic Geometry.* PhD thesis, ETH Zürich, 1999.]

[24] G. Kramer. *Solving Geometric Constraints Systems.* MIT Press, 1992.

[25] G. Kramer. Using degrees of freedom analysis to solve geometric constraint systems. In J. Rossignac and J. Turner, editors, *Symposium on Solid Modeling Foundations and CAD/CAM Applications*, 371-378, Austin, TX, 1991. ACM Press.

[26] G. Kramer. A geometric constraint engine. *Artificial Intelligence*, 58(1-3):327-360, 1992.

[27] G. Kwaiter, V. Gaildrat, and R. Caubet. Interactive constraint system for solid modeling objects. In C. Hoffmann and W. Bronsvoort, editors, Fourth Symposium on Solid Modeling and Applications, 265-270. ACM 1997.

[28] R. S. Latham and A. E. Middleditch. Connectivity analysis: a tool for processing geometric constraints. *Computer Aided Design*, 28(11):917-928, 1996.

[29] J. Y. Lee and K. Kim. Geometric reasoning for knowledge-based parametric design using graph representations. *Computer-Aided Design*, 28(10):831-841, 1996.

[30] W. Leler. *Constraint Programming Languages: Their Specification and Generation.* Addison Wesley, 1988.

[31] J. C. Owen. Algebraic solution for geometry from dimensional constraints. In R. Rossignac and J. Turner, editors, *Symposium on Solid Modeling Foundations and CAD/CAM Applications*, 397-407, Austin, TX, 1991.

[32] M. Sannella. The Sky Blue constraint solver. Technical Report 92-07-02, University of Washington, Dep of Computer Science and Engineering, 1993.

[33] P. Todd. A k-tree generalization that characterizes consistency of dimensioned engineering drawings. *SIAM J. Disc. Math*, 2(2):255-261, 1989.

[34] R. C. Veltkamp and F. Arbab. Geometric constraint propagation with quantum labels. In B. Falcidieno, I. Herman, and C. Pienovi, editors, Eurographics Workshop on Computer Graphics and Mathematics, 211-228. Springer, 1992.

[35] A. Verroust. *Etude de Problemes Lies a la Definition, la Visualisation et l'Animation d'Objects Complexes en Informatique Graphique.* PhD thesis, Universite de Paris-Sud, Centre d'Orsay, 1990.

[36] A. Verroust, F. Schonek, and D. Roller. Rule-oriented method for parameterized computer-aided design. *Computer Aided Design*, 24(10):531-540, 1992.

[37] W.-T. Wu. Basic principles of mechanical theorem proving in geometries. *J. of Systems Sciences and Mathematical Sciences*, 4:207--235, 1986.

[38] W.-T. Wu. Mechanical theorem proving in geometries. In B. Buchberger and G. E. Collins, editors, *Texts and Monographs in Symbolic Computations.* Springer-Verlag, 1994.